Учреждение образования

«Белорусский государственный технологический университет»

Факультет информационных технологий

Кафедра информационных систем и технологий

Специальность 1-40 05 01 «Информационные системы и технологии»

ОТЧЕТ

по учебной ознакомительной практике

Исполнитель

студент 1 курса 2 группы \_\_\_\_\_\_\_\_\_\_\_\_\_ Ободов П.А.

(подпись, дата)

Руководитель практики

от кафедры

Ассистент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Барковский Е.В.

(должность, уч. звание) (подпись, дата)

Отчет защищен с оценкой\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Введение**

Учебная ознакомительная практика является неотъемлемой частью учебного процесса, а также важным этапом в подготовке и становлении будущего специалиста–программиста. Качество подготовки молодых специалистов в образовательных учреждениях находится в прямой зависимости от глубины полученных учащимися знаний в ходе обучения, отработки и закрепления практических навыков. Практика, в свою очередь, играет одну из главных ролей в подготовке и быстрой адаптации студентов и призвана закрепить полученные в университете теоретические знания. В процессе обучения программированию, важное значение имеет практика в создании игр. На сегодняшний момент много компаний которые создают и выпускают игры. Такие как Blizzard, Valve, Activison и многие другие. Причиной актуальности игр является отвлечение внимания от рутины. Были случаи, когда люди рассказывали, что игры помогали им справиться с депрессией. Также качественная постобработка придаёт играм приятную картинку похожую на реальность.В наши дни огромное количество самых разных по интересам людей частенько поигрывают в компьютерные игры. Среди игроков встречаются и бизнесмены, и политики, и домохозяйки, и инженеры, и художники - в целом абсолютно разные люди. Всех их объединяет одно - желание испытать в виртуальных мирах что-то новое, неизведанное, попытать удачу и получить наслаждение как от игрового процесса, так и от достигнутых в игре результатов.

В нашем случае мы хотим рассмотреть практику создания веб-приложения, а точнее интеллектуальной игры «Найди пары карточек». В данном приложении задача участника найти пары карточек быстрее, чем найдёт другой участник.

Цель ознакомительной практики: закрепление и углубление теоретических знаний, полученных по специальным дисциплинам, получение профессиональных навыков работы.

Основные задачи для реализации проекта:

1. Рассмотреть технологию создания компьютерной игры;

2. Сравнить и выбрать для себя приложение для создания игр;

3. Изучить выбранную программу и создать в ней игру;

4. Протестировать игру среди своих друзей и одноклассников;

**Характеристика проекта**

**1.1 Используемые инструменты**

Веб-приложение разрабатывалось в текстовом редакторе Visual Studio Code. Игра может просматриваться в таких браузерах как Internet Explorer, Microsoft Edge, Safari, Opera, Mozilla Firefox, Google Chrome.

Для реализации проекта были выбраны технологи: React JS, CSS3, TypeScript. Структура игры создана с помощью JS, дизайн игры оформлен с помощью CSS. Подробнее про каждый из них:

В отличие от других библиотек JavaScript, предоставляющих полноценную платформу приложений, React ориентируется исключительно на создание представлений приложений через инкапсулированные единицы (называются компонентами), которые сохраняют состояние и генерируют элементы пользовательского интерфейса.

С помощью React разработчики создают веб-приложения, которые изменяют отображение без перезагрузки страницы. Благодаря этому приложения быстро реагируют на действия пользователя, например, заполнение форм, применение фильтров, добавление товаров в корзину и так далее.

React применяют для отрисовки компонентов пользовательского интерфейса. Также библиотека может полностью управлять фронтендом. В этом случае React используют с библиотеками для управления состоянием и роутинга, например, Redux и React Router.

В данном разделе были рассмотрены аналогичные решения, были проанализированы их недостатки и преимущества, для создания в последующем качественного веб-приложения. Были поставлены конкретные задачи для создания интерактивной интеллектуальной игры. Кроме того, были рассмотрены средства реализации программного продукта, такие как языки разметки, а также редактор кода VS Code. Были проанализированы особенности языков и основные дополнительные средства редактора. Все вышеперечисленные пункты понадобятся для достижения поставленной задачи и т.д.

**1.2 Реализация проекта**

Для реализации проекта было принятно решение создать один единый компонент.Как правило, компоненты написаны с использованием JSX, но также есть возможность использования обычного JavaScript. Код примера JSX-формата показан в листинге 1.1.

|  |
| --- |
| return (      <div className="container">    <p className="number-of-strokes">Сделано ходов:{moves}</p>        <div className="cards">          {arrayCards.map((item , index) =>{            let isFlipped = false;            if (openedCards.includes(index)) isFlipped = true            if (matchedCards.includes(item.id)) isFlipped = true            return (              <div key={index} className={`card ${isFlipped ? 'flipped' : ' ' }`}                onClick={flipCard(index)}>                <div className="inner">                  <div className="front">                    <img src={item.img} width="100" alt="front-card" />                  </div>                  <div className="back">                 <img src ={pathIconQuestion} width="100" alt="question mark"/>                  </div>                </div>              </div>            )          })}          </div>          <button className="button-restart" onClick={handleGameRestart}>Начать заново</button>      </div>    );  } |

Листинг 1.1 – Структура JSX-формата

Самым необычным в создании этого веб-приложения является использование хуков. Хуки решают множество, казалось бы, несвязанных между собой, проблем в React, с которыми мы сталкивались в течение пяти лет написания и поддержки десятков тысяч компонентов. Если вы изучаете React, используете его ежедневно или используете другую библиотеку с похожим компонентным подходом, эти проблемы наверняка покажутся вам знакомыми. С помощью хуков вы можете извлечь логику состояния из компонента, чтобы её протестировать или повторно использовать. **Хуки позволяют вам повторно использовать логику состояния, не затрагивая дерево компонентов.** Благодаря этому, хуки легко использовать в разных компонентах и делиться ими с сообществом. Пример использования хуков представлен на листинге 1.2.

|  |
| --- |
| const App = () => {    const [arrayCards , setArrayCards] = useState([])    const [openedCards , setOpenedCards] = useState([])    const [matchedCards , setMatched] = useState([])    const [moves , setMoves] = useState(0)  useEffect(()=>{      setArrayCards(shuffa(pairOfArrayCards))    },[])    const flipCard = (index) => () => {      setOpenedCards(opened => [...opened,index])      setMoves(prevMove => prevMove + 1)    }    useEffect(() =>{      if (openedCards < 2) return      const firstMatched = arrayCards[openedCards[0]];      const secondMatched = arrayCards[openedCards[1]]      if (secondMatched && firstMatched.id === secondMatched.id){        setMatched([...matchedCards, firstMatched.id])      }      if (openedCards.length === 2) setTimeout(() => setOpenedCards([]), 800)    },[openedCards])      const handleGameRestart = () => {      setOpenedCards([])      setMatched([])      setMoves(0)      setArrayCards(shuffa(pairOfArrayCards))    } |

Листинг 1.2 – Структура хуков и их использование

Я решил сделать игру, которая развивает интуицию. Поскольку большая часть игр – пустая трата времени и вызывает проблемы нервной системы. Игра «Найти пары карточек» включает себя карточки расположенные на тыльной стороне. На каждой карточке имеется определенный язык программирования или что-то, что как-то связано с программированием. Задача каждого участника игры перевернуть карточки так, чтобы картинки на них совпали. Карточки расположенные в случайной последовательности. У вас есть возможность открыть две карточки, если картинки на них не совпадают карточки переворачиваются обратно на тыльную сторону через быстрый промежуток времени. При угадывании одинаковых карточек игра продолжается и вам нужно угадать следующую пару карточек. При угадывании шести пар карточек игра закончена и вы победили. В игре имеется кнопка «Начать заново» которая производит сброс игры. За кнопку «Начать заново» отвечает компонент handleGameRestart. Код можно увидеть на листинге 1.3.

|  |
| --- |
| const handleGameRestart = () => {      setOpenedCards([])      setMatched([])      setMoves(0)      setArrayCards(shuffa(pairOfArrayCards))    }   <button className="button-restart" onClick={handleGameRestart}>Начать заново</button> |

Листинг 1.3 – Структура кнопки «Начать заново»

Также сама анимация сделана через CSS3. Процесс открытия и закрытия карточек. Код можно увидеть на листинге 1.4.

|  |
| --- |
| .card.flipped .inner {      transform: rotateY(180deg);  }  .inner{      position: *relative*;      width: 100%;      height: 100%;      transform-style: *preserve-3d*;      transition: 0.6s cubic-bezier(8.38, 0.02, 0.09, 1.66) *all*;  }  .card .front,  .card .back{      display: *flex*;      align-items: *center*;      justify-content: *center*;      border-radius: 8px;      position: *absolute*;      top: 0;      left: 0;      height: 100%;      width: 100%;      backface-visibility: *hidden*; */\*как будто не видим карточки\*/*  }  .card .front{      border: 2px *solid* rgb(214, 214, 26);      background: #fff;      transform: rotateY(180deg);  }  .card .back{      background: #e3b039;      font-size: 100px;      transition: 0.3s *ease* *all*;  }  .card .back:hover,  .button-restart{      background: #dba8b1;      transform: scale(1.02);  }  .button-restart{      cursor: *pointer*;      margin-top: 15px;      font-weight: *bold*;      font-size: 20px;      padding: 10px;      border-radius: 10px;      background: #eabd55;      transition: 0.3s *ease* *all*;  } |

Листинг 1.4 – Структура анимации карточек

Игра протестирована в разных браузерах, таких как Opera, Google Chrome, Microsoft Edge. Пример изображён на рисунке 1.1.



Рисунок 1.1

При совпадении карточек карточки сохраняются и игра продолжается. Продемонстрировано на рисунке 1.2.

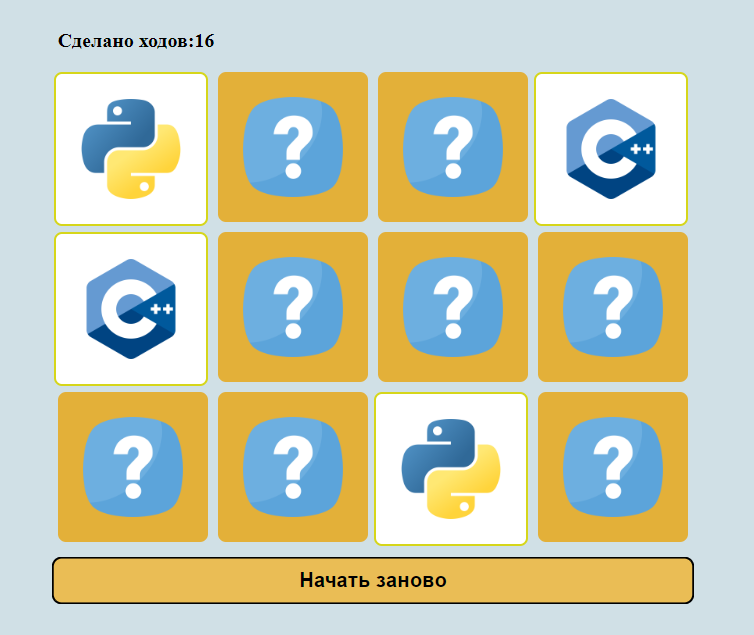


Рисунок 1.2

При совпадении карточек карточки сохраняются и игра продолжается.

**Заключение**

В ходе учебной ознакомительной практики была разработана компьютерная игра «Найди пары карточек» на ReactJS. Выполнены основные пункты технического задания:

* Рассмотреть технологию создания компьютерной игры;
* Сравнить и выбрать для себя приложение для создания игр;
* Продуман алгоритм работы программы, просчитана дальнейшая реализация
* Изучить выбранную программу и создать в ней игру;
* Протестировать игру среди своих друзей и одноклассников;

Была проведена работа с языком программирования JS. Конкретнее с библиотекой ReactJS. Понимание этого языка для меня намного легче, чем других.